# R extRemes Test with LOCA Downscaled CMIP-5 Model Output and some Horesplay with Ensembles (Example Using Precipitation)

Processing for Precipitation Daily Extremes Using LOCA Ensemble Output.

We will be usnig a Generalized Pareto Distribution to target the extreme values and that also requires a threshold to which the method is sensitive. Temperature complicates this since we are looking at a goodly amount of non-stationarity as we move into the future.

A reference for the extRemes package can be found here

Gilleland, E., & Katz, R. (2016). extRemes 2.0: An Extreme Value Analysis Package in R. *Journal of Statistical Software*, **72**(8), 1-39. doi:http://dx.doi.org/10.18637/jss.v072.i08 ( PDF Avaialble at https://www.jstatsoft.org/article/view/v072i08 )

Warning Typos are Legion.

Load Required Libraries

```
library(extRemes)  # extreme data analysis package
```

```
## Loading required package: Lmoments

## Loading required package: distillery

## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'extRemes'

## The following objects are masked from 'package:stats':
##
##     qqnorm, qqplot
```

```
library(ncdf4)     # library for  processing netCDF data
library(lubridate) # library for processing dates and time
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
library(reshape2)  # library for manipulating data frames
library(beanplot)  # library for using bean plots for comparative plots
library(abind)     # library for merging multidimentional arrays
library(stringr)   # library for string manipulation
```

Let's start by identifying specific points, days and ensembles from which to pull data.

It's nice to put this data up top to make changing locatons easier.

```
location_name = "Rapid City"
location_abrv = 'KRAP'
```

```
target_lon  = -104. # degrees east
target_lat  =   44. # degrees north


# location_name = "Brookings"
# location_abrv = 'KBKX';

# target_lon  =  -96.8189167 # degrees east
# target_lat  =  44.3045278 # degrees north
```

And we should include our time periods. Here, we have a baseline period and a future period

```
period_span     = 30.
base_start      = 1976              # start year
base_end        = base_start + period_span-1 # end year  (just before our simulations diverge)

per1_start      = 2020             # start
per1_end        = per1_start + period_span-1 # end yvar2dear
```

And while I am up here I am defining a "wetting rain" even threshold as any rain event exceeding 0.1" per day

I tend to keep this value on reserve when working with rainfall whether I use it or not. (In this case I am not…)

```
wetting_rain_treshold = 0.1 * 25.4 # in -> mm (0.1 in = 2.54 )
```

The LOCA File for Rapid City can be downloaded from

https://github.com/wjcapehart/SDMines_CEE_AES_Math_Ed_Resources/raw/master/Extreme_Event_Assessment/Data_For_Examples/KRAP_LOCA.nc

```
# you will want to change the directory to point to where you are keeping the file.

URL_Root_Directory <- "~/GitHub/SDMines_CEE_AES_Math_Ed_Resources/Extreme_Event_Assessment/Data_For_Exam

# Here is the variable we want to extract
#  these should match the files names since the variable in the file and file name match

target_variable = "prec"
variable_name   = "Total Daily Precip"
variable_units  = "mm" # after post processing (starts as m/s)

threshold_value =  2.0 # This is needed for the Pareto Distribution.

# get the extreme function to use

extreme_func = "GP"


# get URL files
LOCA_File  = paste(location_abrv,
                   "_LOCA.nc",
                   sep="")


# make the final URLs for extracting with the "paste" function
```

```r
LOCA_URL = paste(URL_Root_Directory,  # string 1 to concatenate
                 LOCA_File,          # string 2 to concatenate
                 sep=""               # separation character ("" = none)
                )
```

We now open the files…

the output of these functions are "handles" by which we can reference the file

```r
nc.loca <- nc_open(filename = LOCA_URL)
```

To view the inventory (or "metadata") of a netCDF file you can "print()" the metadata

Let's do this with the RCP 8.5 file

```r
print(nc.loca)
```

```
## File ~/GitHub/SDMines_CEE_AES_Math_Ed_Resources/Extreme_Event_Assessment/Data_For_Examples/KRAP_LOCA
##
##      13 variables (excluding dimension variables):
##         double latitude[]   (Contiguous storage)
##             axis: Y
##             units: degrees_north
##             short_name: latitude
##             long_name: Latitude of Grid Point
##             description: Latitude of Grid Point
##         double longitude[]   (Contiguous storage)
##             axis: X
##             units: degrees_west
##             short_name: longitude
##             long_name: Longitude of Grid Point
##             description: Longitude of Grid Point
##         string station_name[]   (Contiguous storage)
##             long_name: Station Name
##             description: Station Name
##         string station_icao[]   (Contiguous storage)
##             long_name: Station ICAO Designator
##             description: Station ICAO Designator
##         float latitude_station[]   (Contiguous storage)
##             units: degrees_north
##             short_name: latitude
##             long_name: Latitude of Station
##             description: Latitude of Station
##         float longitude_station[]   (Contiguous storage)
##             units: degrees_east
##             short_name: longitude
##             long_name: Longitude of Station
##             description: Longitude of Station
##         int available_inventory[scenario_list,ensemble_model,variable_list]   (Contiguous storage)
##             comment: 1 = available ;  0 = not available
##             short_name: Inventory of LOCA Scenarios
##             description: Inventory of LOCA Scenarios
##             _FillValue: -2147483647
##         float tmax_hist[ensemble_model,time_hist]   (Contiguous storage)
##             coordinates: time_hist latitude longitude
##             cell_methods: tmax_hist:maximum
```

```
##            units: K
##            standard_name: air_temperature
##            long_name: Maximum Daily Temperature (Historical)
##            description: Maximum Daily Temperature (Historical)
##            _FillValue: 9.96920996838687e+36
##        float tmin_hist[ensemble_model,time_hist]   (Contiguous storage)
##            _FillValue: 9.96920996838687e+36
##            description: Maximum Daily Temperature (Historical)
##            long_name: Maximum Daily Temperature (Historical)
##            standard_name: air_temperature
##            units: K
##            cell_methods: tmax_hist:maximum
##            coordinates: time_hist latitude longitude
##        float prec_hist[ensemble_model,time_hist]   (Contiguous storage)
##            _FillValue: 9.96920996838687e+36
##            description: Total Daily Precipitation (Historical)
##            long_name: Total Daily Precipitation (Historical)
##            standard_name: precipitation_amount
##            units: kg m-2
##            cell_methods: tmax_hist:sum
##            coordinates: time_hist latitude longitude
##        float tmax_futr[RCP_scenario,ensemble_model,time_futr]   (Contiguous storage)
##            coordinates: time_futr latitude longitude
##            cell_methods: tmax_futr:maximum
##            units: K
##            standard_name: air_temperature
##            long_name: Maximum Daily Temperature (Projected)
##            description: Maximum Daily Temperature (Projected)
##            _FillValue: 9.96920996838687e+36
##        float tmin_futr[RCP_scenario,ensemble_model,time_futr]   (Contiguous storage)
##            _FillValue: 9.96920996838687e+36
##            description: Maximum Daily Temperature (Projected)
##            long_name: Maximum Daily Temperature (Projected)
##            standard_name: air_temperature
##            units: K
##            cell_methods: tmax_futr:maximum
##            coordinates: time_futr latitude longitude
##        float prec_futr[RCP_scenario,ensemble_model,time_futr]   (Contiguous storage)
##            _FillValue: 9.96920996838687e+36
##            description: Total Daily Precipitation (Projected)
##            long_name: Total Daily Precipitation (Projected)
##            standard_name: precipitation_amount
##            units: kg m-2
##            cell_methods: tmax_futr:sum
##            coordinates: time_futr latitude longitude
##
##     6 dimensions:
##        time_hist  Size:20454
##            description: Time (Historical Period)
##            units: days since 1900-01-01 00:00:00
##            long_name: Time (Historical Period)
##            calendar: standard
##            missing_value: 1.00000001504747e+30
##            axis: T
```

```
##             standard_name: time
##             _FillValue: 1.00000001504747e+30
##             _ChunkSizes: 1
##             bounds: time_bnds
##         time_futr  Size:34698
##             description: Time (Projected Period)
##             units: days since 1900-01-01 00:00:00
##             long_name: Time (Projected Period)
##             calendar: standard
##             missing_value: 1.00000001504747e+30
##             axis: T
##             standard_name: time
##             _FillValue: 1.00000001504747e+30
##             _ChunkSizes: 1
##             bounds: time_bnds
##         ensemble_model  Size:31
##             long_name: Ensemble Model Member
##             description: Ensemble Model Member
##         RCP_scenario  Size:2
##             long_name: CMIP-5 Representative Concentration Pathway Scenarios
##             description: CMIP-5 Representative Concentration Pathway Scenarios
##         scenario_list  Size:3
##             short_name: List of Scenarios for Inventory
##             description: List of Scenarios for Inventory
##         variable_list  Size:3
##             description: List of Variables for Inventory
##             short_name: List of Variables for Inventory
##
##     23 global attributes:
##         title: Historical LOCA Statistical Downscaling (Localized Constructed Analogs) Statistically
##         creator_name: Original: David Piercem (OUI-USGS);
## Modified: Bill Capehart SD School of Mines
##         creator_email: Original: dpierce@ucsd.edu;
## Modified: William.Capehart@sdsmt.edu
##         institution: Original: USGS Office of Water Information;
## Modified: SD School of Mines and Technology
##         Conventions: CF-1.4
##         project: Comparative Analysis of Downscaled Climate Simulations, Providing Guidance to End U
##         cdm_data_type: Station
##         summary: LOCA is a statistical downscaling technique that uses past history to add improved 
## We have used LOCA to downscale 32 global climate models from the CMIP5 archive at a 1/16th degree sp
## The historical period is 1950-2005, and there are two future scenarios available: RCP 4.5 and RCP 8.
## The variables currently available are daily minimum and maximum temperature, and daily precipitation
## For more information visit: http://loca.ucsd.edu/
##         acknowledgment: Pierce, D. W., D. R. Cayan, and B. L. Thrasher, 2014: Statistical downscalin
## We acknowledge the World Climate Research Programme's Working Group on Coupled Modeling, which is re
## For CMIP, the U.S. Department of Energy's Program for Climate Model Diagnosis and Intercomparison pr
##         comment: Original Prioduct Extracted to a Single Station Point, Original Prioduct Calibrated
##         history: Dataset was deflated and aggregated for publication by the USGS by dblodgett@usgs.g
## Dataset was extracted for a single point and converted to common SI units by William.Capehart@sdsmt.
##         time_coverage_start: 1950-01-01T00:00
##         time_coverage_end: 2100-12-31T00:00
##         station_name: Rapid City, SD
##         station_icao: KRAP
```

```
##          latitude_station: 44.0453338623047
##          longitude_station: 256.942626953125
##          geospatial_lat_min: 44.03125
##          geospatial_lat_max: 44.03125
##          geospatial_lon_min: 256.96875
##          geospatial_lon_max: 256.96875
##          keywords: precipitation, temperature
##          license: Freely available
```

Now for Time Control the RCP 8.5 runs go from 1920-2100 and have 2172 monthly time steps the RCP 4.5 runs go from 1920-2080 and have 1932 monthly time steps

Our data uses a 365-day year (no leap years) so we will doing this brute force.

We are also only going to 2080.

```r
# get time values

time_hist<-ncvar_get(nc.loca, "time_hist")
tunits<-ncatt_get(nc.loca, "time_hist", attname="units")
tustr<-strsplit(tunits$value, " ")

time_reference_point_hist = unlist(tustr)[3]


time_futr <-ncvar_get(nc.loca, "time_futr")
tunits    <-ncatt_get(nc.loca, "time_futr", attname="units")
tustr     <-strsplit(tunits$value, " ")

# and tidy things up

time_reference_point_futr = unlist(tustr)[3]
```

Now for the easier coordinates to access

```r
# create ensemble dimension

ensemble <- ncvar_get(nc.loca, "ensemble_model")

ensembles_to_use = ensemble[!str_detect(ensemble, "CNRM")]
ensembles_to_use = ensembles_to_use[!str_detect(ensembles_to_use, "GISS")]

n_ensembles = length(ensembles_to_use)

print(ensembles_to_use)
```

```
##  [1] "ACCESS1-0_r1i1p1"      "ACCESS1-3_r1i1p1"
##  [3] "CCSM4_r6i1p1"          "CESM1-BGC_r1i1p1"
##  [5] "CESM1-CAM5_r1i1p1"     "CMCC-CMS_r1i1p1"
##  [7] "CMCC-CM_r1i1p1"        "CSIRO-Mk3-6-0_r1i1p1"
##  [9] "CanESM2_r1i1p1"        "FGOALS-g2_r1i1p1"
## [11] "GFDL-CM3_r1i1p1"       "GFDL-ESM2G_r1i1p1"
## [13] "GFDL-ESM2M_r1i1p1"     "HadGEM2-AO_r1i1p1"
## [15] "HadGEM2-CC_r1i1p1"     "HadGEM2-ES_r1i1p1"
## [17] "IPSL-CM5A-LR_r1i1p1"   "IPSL-CM5A-MR_r1i1p1"
## [19] "MIROC-ESM-CHEM_r1i1p1" "MIROC-ESM_r1i1p1"
## [21] "MIROC5_r1i1p1"         "MPI-ESM-LR_r1i1p1"
```

```
## [23] "MPI-ESM-MR_r1i1p1"     "MRI-CGCM3_r1i1p1"
## [25] "NorESM1-M_r1i1p1"      "bcc-csm1-1-m_r1i1p1"
```

```
# import spatial coordinates

scenario <- ncvar_get(nc.loca, "RCP_scenario")
print(scenario)
```

```
## [1] "RCP 4.5" "RCP 8.5"
```

```
scenario = c("RCP45","RCP85")
```

Pull Variable

```
varid.hist = paste(target_variable,
                   "_hist",
                   sep="")

varid.futr = paste(target_variable,
                   "_futr",
                   sep="")

var2d.hist = ncvar_get(nc      = nc.loca,                      # netcdf file ID
                       varid   = varid.hist,            # variable name from file
                       verbose = FALSE                      # print diagnostic data
                       )                  # scaling temperature from K to DegC

var2d.futr = ncvar_get(nc    = nc.loca,                      # netcdf file ID
                       varid   = varid.futr,            # variable name from file
                       verbose = FALSE                      # print diagnostic data
                       )                   # scaling temperature from K to DegC
```

... Then we pull the RCP 8.5 scenario...

Assign dimension names to arrays. these are strings including things that shoudl be numbers.

```
dimnames(var2d.hist) <- list(ensemble,time_hist)

dimnames(var2d.futr) <- list(scenario,ensemble,time_futr)

size.hist2 = 2*20454*31
var2d.hist2  <- array(1:size.hist2, dim=c(2,31,20454))

var2d.hist2[1,,] = var2d.hist
var2d.hist2[2,,] = var2d.hist

dimnames(var2d.hist2) <- list(scenario,ensemble,time_hist)


var2d.hist = var2d.hist2
remove(var2d.hist2)

var3d = abind(var2d.hist, var2d.futr, along=3)
```

Now we flatten all of our fields into a single list or data frame by “flattening” the 2-D time/ensemble aray to a single list with the melt command

```
variable = melt(data       = var3d,              # your array
              na.rm        = TRUE,                  # don't use missing values
              varnames     = c("Scenario","Ensemble","Time"),  # names of your two dimensions
              value.name   = "Variable")                # the final name of your aray value

variable$Time = as.Date(variable$Time, origin=time_reference_point_hist)
```

And we tidy things up once again

```
remove(nc.loca,
       varid.hist,
       varid.futr,
       URL_Root_Directory,
       LOCA_File,
       LOCA_URL,
       tunits,
       tustr,
       var3d,
       var2d.futr,
       var2d.hist)
```

We can also add some utility vectors to break things down by month, year and decade

```
variable$month   = month(variable$Time)
variable$year    = year(variable$Time)
variable$day    = day(variable$Time)
variable$decade = trunc( variable$year/10. )*10
```

And we'll limit ourselves to the period between May and September

```
#  variable = variable[(variable$month>=5 & variable$month<=9),]
```

It's not a bad idea to review the content of your data frame…

```
str(variable)
```

```
## 'data.frame':    3157908 obs. of  8 variables:
##  $ Scenario: Factor w/ 2 levels "RCP45","RCP85": 1 2 1 2 1 2 1 2 1 2 ...
##  $ Ensemble: Factor w/ 31 levels "ACCESS1-0_r1i1p1",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ Time    : Date, format: "1950-01-01" "1950-01-01" ...
##  $ Variable: num  0.4 0.4 1.3 1.3 0 ...
##  $ month   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ year    : num  1950 1950 1950 1950 1950 1950 1950 1950 1950 1950 ...
##  $ day     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ decade  : num  1950 1950 1950 1950 1950 1950 1950 1950 1950 1950 ...
```

Now we're ready to do the analysis part.

we can break down the procedure to

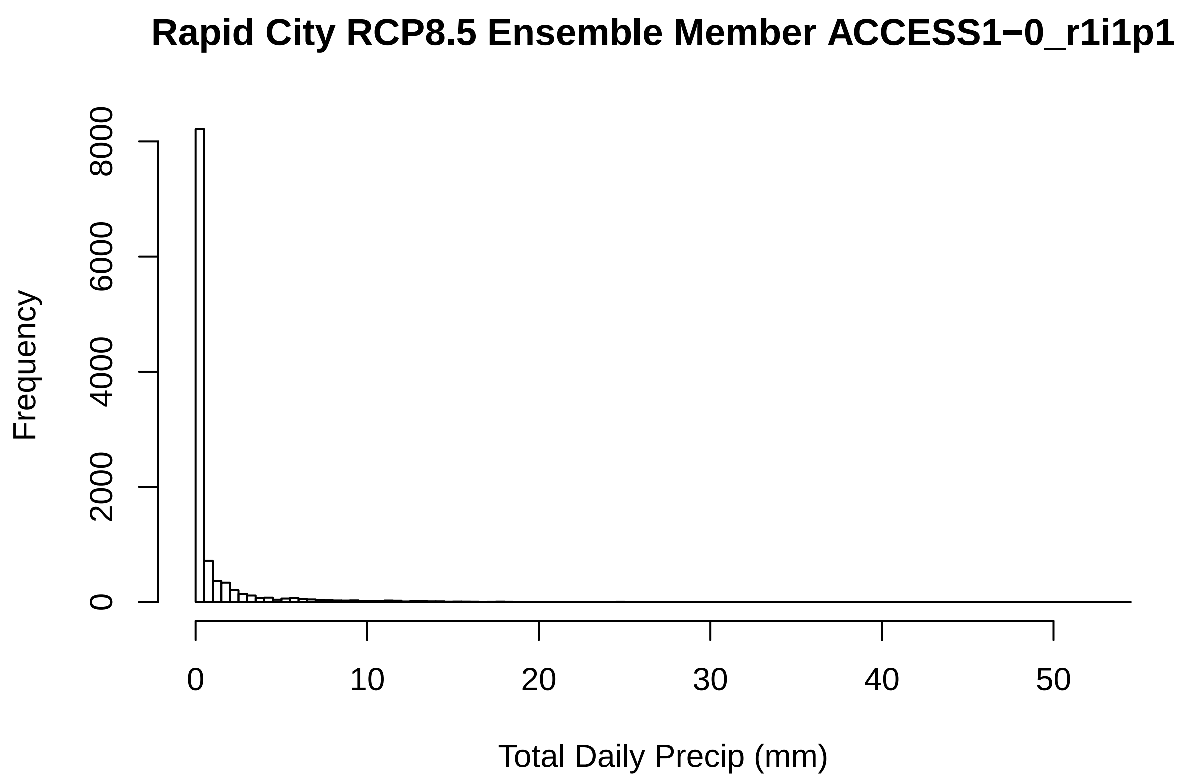
1) par down the data vector by time period and ensemble number.
2) run and save our stats.

Here is a working sample template to get things started.

First we start extract our subset.

```
ens_target = ensembles_to_use[1] # just use the first ensemble member
rpc_target = "RCP45"
subset = subset(variable,
```

```r
                 (Scenario==rpc_target)  &
                 (Ensemble==ens_target)  &
                 ((year>=base_start) &
                  (year<=base_end)    )
                 )

# always nice to see what your data looks like.

hist(x      = subset$Variable, # omitting zero temperatures days
     xlab   = paste(variable_name,                   # xaxis title
                    " (",
                    variable_units,
                    ")",
                    sep=""),
     main   = paste(location_name,
                    paste("RCP8.5 Ensemble Member",ens_target),
                    sep = " "),
     freq   = TRUE,
     breaks = 100
     )
```




Now we run the fit command.

We will need a minimum threshold or the method fails.

Chose a reasonable value, here we chose 2mm (0.08“)

For reference, a wetting rain event is 0.1" or 2.54 mm

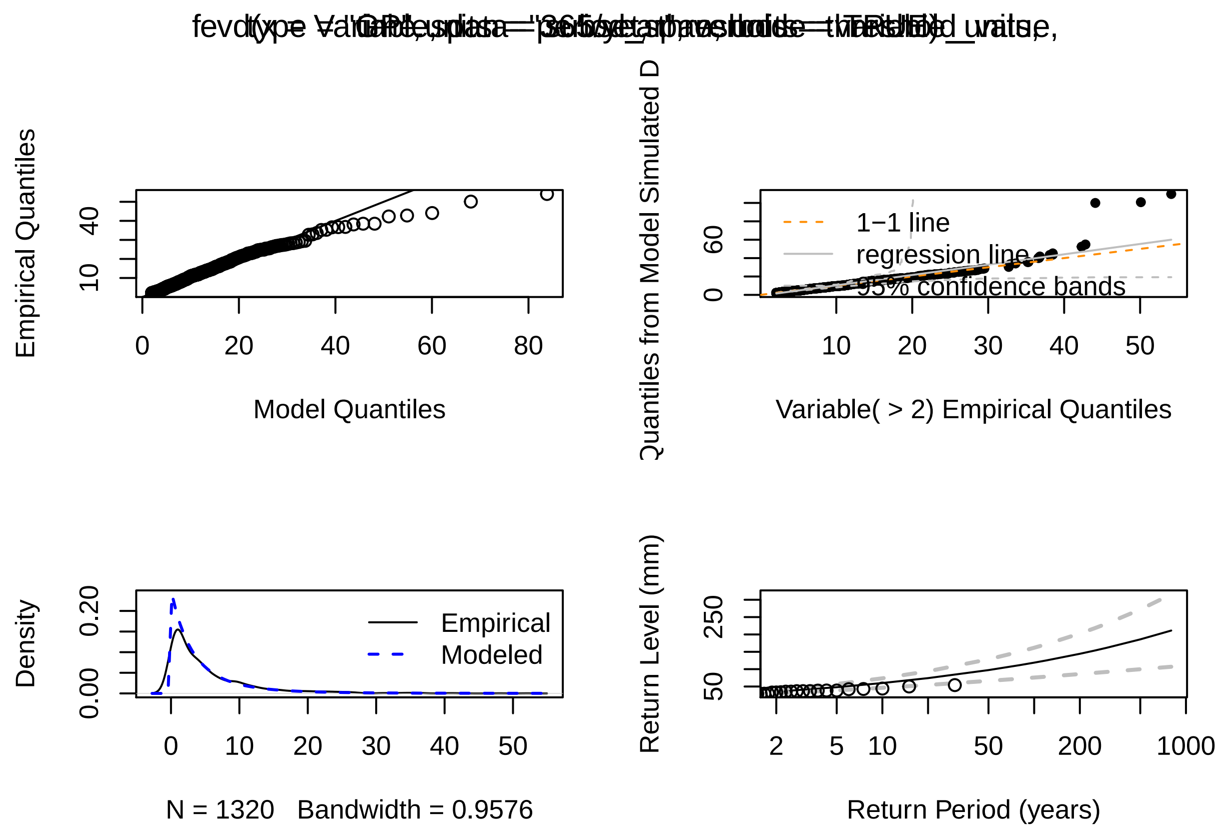
We are also using the Generalized Pareto method.

```
fit_GP  = fevd(x          = Variable,
               data       = subset,
               verbose    = TRUE,
               threshold  = threshold_value,
               units      = variable_units,
               time.units = "365/year",
               type       = "GP",
               span       = period_span
               )
```

```
##
##  Data span  0.08219178 years
## Setting up parameter model design matrices.
## Parameter model design matrices set up.
## Using Lmoments estimates as initial estimates.  Initial value = 3479.568
## Initial estimates are:
##     scale     shape
## 3.9266053 0.2542597
## Beginning estimation procedure.
## initial  value 3479.567585
## final  value 3479.484097
## converged
## Time difference of 0.4866929 secs
```

And we plot the results. You would prefer for the lines to rest on the “1:1” through a reasonable period. For a 30 year span, I’ll consider 15 return yers to be acceptable.

```
plot(fit_GP)
```




We can also extract specific return events.

```
return_intervals = seq(from =             2,
                       to   = period_span/2,
                       by   =             1)


n_return_intervals = length(return_intervals)

return_GP = return.level(x             = fit_GP,
                         return.period = return_intervals,
                         do.ci         = TRUE)

print(return_GP)
```

```
## fevd(x = Variable, data = subset, threshold = threshold_value,
##     type = "GP", span = period_span, units = variable_units,
##     time.units = "365/year", verbose = TRUE)
##
## [1] "Normal Approx."
##
##                        95% lower CI Estimate 95% upper CI
## 2-year return level        30.14403 35.24617     40.34830
## 3-year return level        33.89638 40.56405     47.23171
## 4-year return level        36.69425 44.68453     52.67480
## 5-year return level        38.94338 48.09521     57.24704
## 6-year return level        40.83282 51.02952     61.22623
```

```
## 7-year return level      42.46689 53.61903     64.77117
## 8-year return level      43.90961 55.94585     67.98209
## 9-year return level      45.20321 58.06500     70.92679
## 10-year return level     46.37711 60.01529     73.65346
## 11-year return level     47.45265 61.82520     76.19775
## 12-year return level     48.44584 63.51634     78.58685
## 13-year return level     49.36900 65.10550     80.84200
## 14-year return level     50.23183 66.60600     82.98016
## 15-year return level     51.04212 68.02863     85.01513
```

```
# tidy up the demo

remove(return_GP,
       fit_GP,
       subset)
```

To implement. Let's create the storage arrays We will copy this over and over.

```
# for each case/period create arrays for the return and CI bounds

# first for our base period

rcp45_base_period_return  = array(data= 0,
                                  dim = c(n_ensembles,
                                          n_return_intervals))

dimnames(rcp45_base_period_return) = list(ensembles_to_use,
                                         return_intervals)

rcp45_base_period_ci_ub   = rcp45_base_period_return
rcp45_base_period_ci_lb   = rcp45_base_period_return

rcp85_base_period_return  = rcp45_base_period_return
rcp85_base_period_ci_ub   = rcp45_base_period_return
rcp85_base_period_ci_lb   = rcp45_base_period_return

# then for our test period(s)

rcp45_per1_period_return  = rcp45_base_period_return
rcp45_per1_period_ci_ub   = rcp45_base_period_return
rcp45_per1_period_ci_lb   = rcp45_base_period_return

rcp85_per1_period_return  = rcp45_base_period_return
rcp85_per1_period_ci_ub   = rcp45_base_period_return
rcp85_per1_period_ci_lb   = rcp45_base_period_return
```

create a loop that goes between the ensembles.

```
ens_target_num = 0

for (ens_target in ensembles_to_use) {
  ens_target_num = ens_target_num + 1

  ######################################
  #
  # Base Period
```

```
#

# create segments for RCP 45 and RCP 85 for your period(s)


ens_target

rpc_target = "RCP45"

subset = subset(variable,
                (Ensemble==ens_target)  &
                (Scenario==rpc_target)  &
                ((year>=base_start) &
                   (year<=base_end)   )
                     )

# fit an extreme value dist to the data

fit_GP_45 = fevd(x          = Variable,
                 data       = subset,
                 verbose    = FALSE,
                 threshold  = threshold_value,
                 units      = variable_units,
                 time.units = "365/year",
                 type       = "GP",
                 span       = period_span
                 )


rpc_target = "RCP85"

subset = subset(variable,
                (Ensemble==ens_target)  &
                (Scenario==rpc_target)  &
                ((year>=base_start) &
                   (year<=base_end)   )
                     )



fit_GP_85 = fevd(x          = Variable,
                 data       = subset,
                  verbose    = FALSE,
                 threshold  = threshold_value,
                 units      = variable_units,
                 time.units = "365/year",
                 type       = "GP",
                 span       = period_span
                 )

# calculate the return periods and load into the temporary storage arrays.

return_GP = return.level(x            = fit_GP_45,
```

```
                             return.period = return_intervals,
                             do.ci         = TRUE)

rcp45_base_period_ci_lb[ ens_target_num, ] = return_GP[ , 1]
rcp45_base_period_return[ens_target_num, ] = return_GP[ , 2]
rcp45_base_period_ci_ub[ ens_target_num, ] = return_GP[ , 3]


return_GP = return.level(x             = fit_GP_85,
                             return.period = return_intervals,
                             do.ci         = TRUE)

rcp85_base_period_ci_lb[ ens_target_num, ] = return_GP[ , 1]
rcp85_base_period_return[ens_target_num, ] = return_GP[ , 2]
rcp85_base_period_ci_ub[ ens_target_num, ] = return_GP[ , 3]

# tidy up

remove(subset,
       fit_GP_45,
       fit_GP_85,
       return_GP)

#
######################################



######################################
#
#  Period #1
#

# create segments for RCP 45 and RCP 85 for your period(s)


rpc_target = "RCP45"

subset = subset(variable,
                (Ensemble==ens_target)  &
                (Scenario==rpc_target)  &
                ((year>=per1_start) &
                   (year<=per1_end)   )
                     )

# fit an extreme value dist to the data

fit_GP_45 = fevd(x          = Variable,
                 data       = subset,
                 verbose    = FALSE,
                 threshold  = threshold_value,
                 units      = variable_units,
                 time.units = "365/year",
```

```
                  type       = "GP",
                  span       = period_span
                  )

  rpc_target = "RCP85"

  subset = subset(variable,
                 (Ensemble==ens_target)  &
                 (Scenario==rpc_target)  &
                  ((year>=per1_start) &
                    (year<=per1_end)   )
                      )

  fit_GP_85 = fevd(x          = Variable,
                  data        = subset,
                  verbose     = FALSE,
                  threshold   = threshold_value,
                  units       = variable_units,
                  time.units  = "365/year",
                  type        = "GP",
                  span        = period_span
                  )

  # calculate the return periods and load into the temporary storage arrays.

  return_GP = return.level(x             = fit_GP_45,
                          return.period = return_intervals,
                          do.ci         = TRUE)

  rcp45_per1_period_ci_lb[ ens_target_num, ] = return_GP[ , 1]
  rcp45_per1_period_return[ens_target_num, ] = return_GP[ , 2]
  rcp45_per1_period_ci_ub[ ens_target_num, ] = return_GP[ , 3]


  return_GP = return.level(x             = fit_GP_85,
                          return.period = return_intervals,
                          do.ci         = TRUE)

  rcp85_per1_period_ci_lb[ ens_target_num, ] = return_GP[ , 1]
  rcp85_per1_period_return[ens_target_num, ] = return_GP[ , 2]
  rcp85_per1_period_ci_ub[ ens_target_num, ] = return_GP[ , 3]

  # tidy up

  remove(subset,
         fit_GP_45,
         fit_GP_85,
         return_GP)

  #
  ######################################

}
```

Convert output into neat and tidy data frames

```
##################################################
#
# let's start with the base period
#

# melt 2-d arrays into frames.

rcp45_base   = melt(data       = rcp45_base_period_return,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Return_Period")

rcp45_cib_lb = melt(data       = rcp45_base_period_ci_lb,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Lower_95_CI")

rcp45_cib_ub = melt(data       = rcp45_base_period_ci_ub,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Upper_95_CI")




rcp85_base   = melt(data       = rcp85_base_period_return,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP85_Return_Period")

rcp85_cib_lb = melt(data       = rcp85_base_period_ci_lb,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP85_Lower_95_CI")

rcp85_cib_ub = melt(data       = rcp85_base_period_ci_ub,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP85_Upper_95_CI")



# combine the fields into a single data frame for each one.

rcp45_base$RCP45_Lower_95_CI = rcp45_cib_lb$RCP45_Lower_95_CI
rcp45_base$RCP45_Upper_95_CI = rcp45_cib_ub$RCP45_Upper_95_CI

rcp85_base$RCP85_Lower_95_CI = rcp85_cib_lb$RCP85_Lower_95_CI
rcp85_base$RCP85_Upper_95_CI = rcp85_cib_ub$RCP85_Upper_95_CI


# tidy up
```

```
remove(rcp85_base_period_ci_lb,
       rcp85_base_period_return,
       rcp85_base_period_ci_ub,
       rcp85_cib_lb,
       rcp85_cib_ub)

remove(rcp45_base_period_ci_lb,
       rcp45_base_period_return,
       rcp45_base_period_ci_ub,
       rcp45_cib_lb,
       rcp45_cib_ub)


#
################################################

################################################
#
# and we can continue with the test period
#


rcp45_per1   = melt(data       = rcp45_per1_period_return,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Return_Period")

rcp45_ci1_lb = melt(data       = rcp45_per1_period_ci_lb,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Lower_95_CI")

rcp45_ci1_ub = melt(data       = rcp45_per1_period_ci_ub,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP45_Upper_95_CI")




rcp85_per1   = melt(data       = rcp85_per1_period_return,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP85_Return_Period")

rcp85_ci1_lb = melt(data       = rcp85_per1_period_ci_lb,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
                    value.name = "RCP85_Lower_95_CI")

rcp85_ci1_ub = melt(data       = rcp85_per1_period_ci_ub,
                    varnames   = c("Ensembles","return_intervals"),
                    na.rm      = TRUE,
```

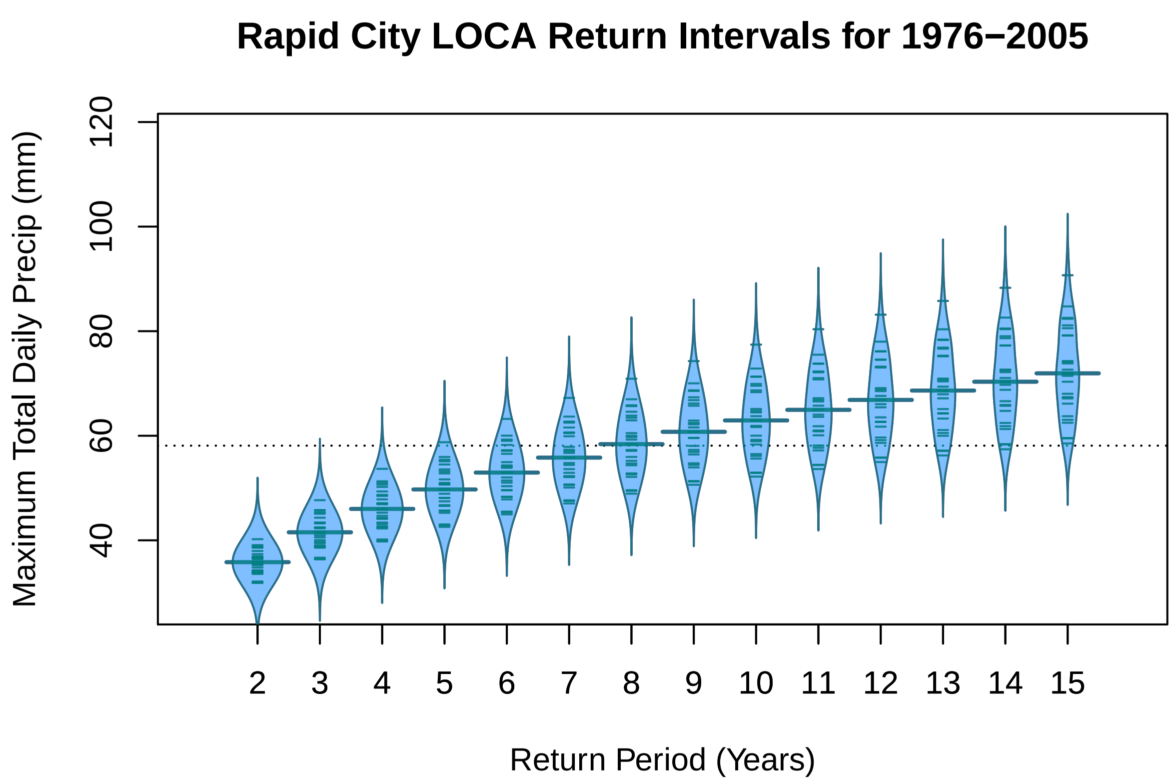
```
                    value.name = "RCP85_Upper_95_CI")



# combine the fields into a single data frame for each one.

rcp45_per1$RCP45_Lower_95_CI = rcp45_ci1_lb$RCP45_Lower_95_CI
rcp45_per1$RCP45_Upper_95_CI = rcp45_ci1_ub$RCP45_Upper_95_CI

rcp85_per1$RCP85_Lower_95_CI = rcp85_ci1_lb$RCP85_Lower_95_CI
rcp85_per1$RCP85_Upper_95_CI = rcp85_ci1_ub$RCP85_Upper_95_CI


# tidy up

remove(rcp85_per1_period_ci_lb,
       rcp85_per1_period_return,
       rcp85_per1_period_ci_ub,
       rcp85_ci1_lb,
       rcp85_ci1_ub)

remove(rcp45_per1_period_ci_lb,
       rcp45_per1_period_return,
       rcp45_per1_period_ci_ub,
       rcp45_ci1_lb,
       rcp45_ci1_ub)


#
################################################
```

Plot return output

```
# I am making my own colours

darkcyan = rgb(red   = 0.00,
               green = 0.50,
               blue  = 0.50,
               alpha = 0.75)  # 1 = opaque; 0 = fully clear

cyan     = rgb(red   = 0.00,
               green = 1.00,
               blue  = 1.00,
               alpha = 0.50)

darkblue = rgb(red   = 0.00,
               green = 0.00,
               blue  = 0.50,
               alpha = 0.75)

blue     = rgb(red   = 0.00,
               green = 0.00,
               blue  = 1.00,
               alpha = 0.50)
```

```
darkmag  = rgb(red   = 0.50,
               green = 0.00,
               blue  = 0.50,
               alpha = 0.75)

magenta  = rgb(red   = 1.00,
               green = 0.00,
               blue  = 1.00,
               alpha = 0.50)


darkred  = rgb(red   = 0.50,
               green = 0.00,
               blue  = 0.00,
               alpha = 0.75)

red      = rgb(red   = 1.00,
               green = 0.00,
               blue  = 0.00,
               alpha = 0.50)

beanplot(formula     = RCP85_Return_Period~return_intervals,  # formula selection for y axis
         data        = rcp85_base,                            # data frame to use
         col         = c(magenta,  # area fill                # Color Scheme
                         darkmag,  # lines
                         darkmag,  # outside of bean line
                         darkmag), # mean line
         border      = darkmag,                               # border color
         overallline = "median",                              # can't get rid of this dang thing
         beanlines   = "median",                              # use median for the "central value"
         main        = paste(location_name,
                             " LOCA Return Intervals for ",         # title caption
                             base_start,
                            "-",
                            base_end,
                            sep=""),
         xlab        = "Return Period (Years)",               # xaxis title
         ylab        = paste("Maximum ",
                             variable_name,                   # yaxis title
                             " (",
                             variable_units,
                             ")",
                             sep=""),
         log         = "",                                    # overide log axis on y (it defaults to l
         xlim        = c(0,period_span/2),                    # x axis range
         ylim        = c(min(rcp45_base$RCP45_Lower_95_CI,    # y axis range
                             rcp85_base$RCP85_Lower_95_CI,
                             rcp45_per1$RCP45_Lower_95_CI,
                             rcp85_per1$RCP45_Lower_95_CI),
                         max(rcp45_base$RCP45_Upper_95_CI,    # y axis range
                             rcp85_base$RCP85_Upper_95_CI,
                             rcp45_per1$RCP45_Upper_95_CI,
```

```
                              rcp85_per1$RCP45_Upper_95_CI)
                         )
        )




beanplot(formula     = RCP45_Return_Period~return_intervals,
         data         = rcp45_base,
         add          = TRUE,
         beanlines    = "median",
         overallline = "median",
         col          = c(cyan,       # area fill
                           darkcyan,  # lines
                           darkcyan,  # outside of bean line
                           darkcyan), # mean line
         border       = darkcyan
        )
```




Trying something new. Now we can do this a "split" plot.

and for the future period.;;

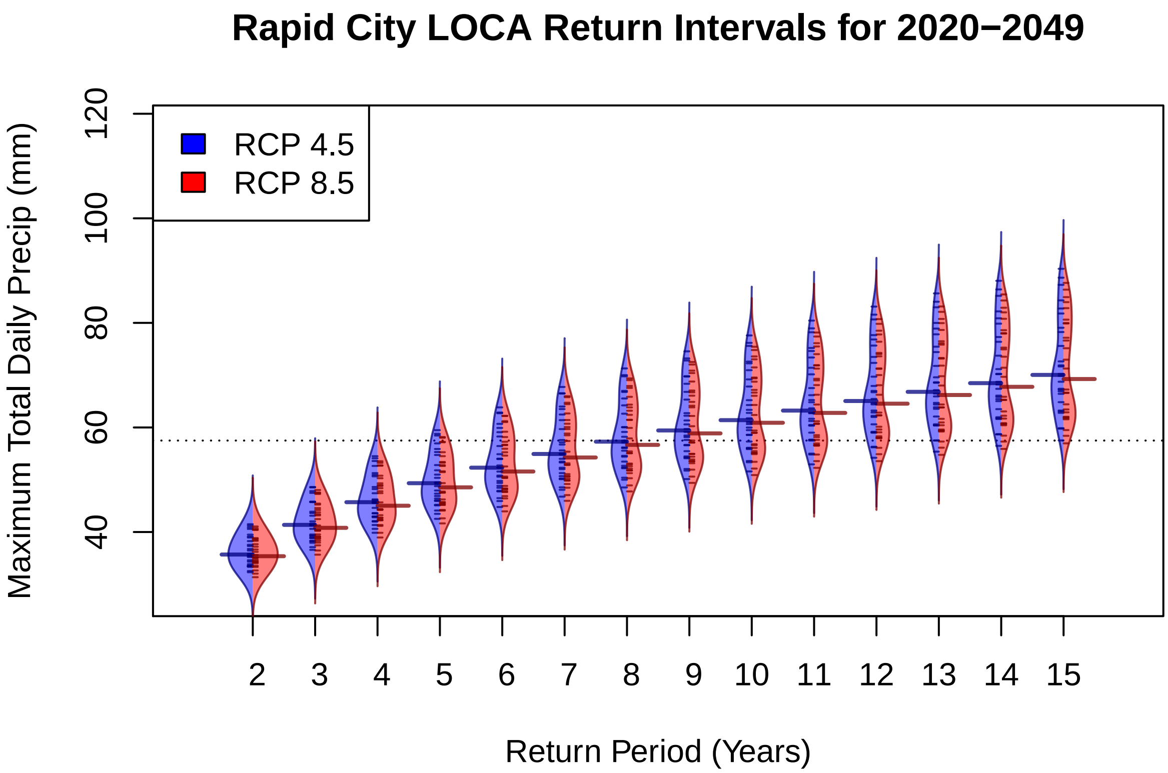
```
# create a temporary array to hold just the return periods for simplicity

deleteme     = data.frame(Ensembles        = rcp45_base$Ensembles,
                          return_intervals = rcp45_base$return_intervals,
```

```r
                          RCP45          = rcp45_per1$RCP45_Return_Period,
                          RCP85          = rcp85_per1$RCP85_Return_Period)

# melt the RCP45 and RCP85 columns into a new field.
base_all    = melt(data       = deleteme,
                   varnames   = c("Ensembles","return_intervals","Scenario"),
                   id.vars    = c("Ensembles","return_intervals"),
                   na.rm      = TRUE,
                   value.name = "Return_Period_Values")

colnames(base_all) <- c("Ensembles","return_intervals","Scenario","Return_Period_Values")

# for this to work below, the program will need to search through ONE variable for the axis
# you can use two (or more) sub-categories.
#
# to do this you need to create a new variable in the data table that will have two
# parts, separated by a space character.
#
# this your x-axis + "series" category would be
#
#  001 Cat1
#  002 Cat1
#  003 Cat1
#  001 Cat2
#  002 Cat3
# ... and so on.
#
# in this case we will have our return period year and then the RCP scenario.
# (We could also do this for different periods for a single RCP scenario)
#
# thus...
#  02 RCP45 (we need that leading zero)
#  02 RCP85
#   ...
#  20 RCP45
#  20 RCP85


base_all$merged_period_scenarios = paste(sprintf("%2d", base_all$return_intervals),
                                         base_all$Scenario,
                                         sep = " ")

# tidy up!

remove(deleteme)
```

And now we can plot this one. This will create a "split" bean plot which is about as complex and messy as I want to get right now.

```r
beanplot(formula     = Return_Period_Values~merged_period_scenarios,  # formula selection for y axis
         data        = base_all,                                     # data frame to use
         col         = list(c(blue, darkblue,darkblue,darkblue),   # same order as earlier for each seri
                                c(red,  darkred,  darkred, darkred)),
         border      = c(darkblue,darkred),
```

```
         overallline = "median",                          # can't get rid of this dang thing
         beanlines   = "median",                          # use median for the "central value"
         side        = "both",
         main        = paste(location_name,
                             " LOCA Return Intervals for ",         # title caption
                             per1_start,
                             "-",
                             per1_end,
                             sep=""),
         xlab        = "Return Period (Years)",           # xaxis title
         ylab        = paste("Maximum ",
                             variable_name,               # yaxis title
                             " (",
                             variable_units,
                             ")",
                             sep=""),
         log         = "",                                # overide log axis on y (it defaults to l
         xlim        = c(0,period_span/2),                # x axis range
         ylim        = c(min(rcp45_base$RCP45_Lower_95_CI,    # y axis range
                             rcp85_base$RCP85_Lower_95_CI,
                             rcp45_per1$RCP45_Lower_95_CI,
                             rcp85_per1$RCP45_Lower_95_CI),
                         max(rcp45_base$RCP45_Upper_95_CI,    # y axis range
                             rcp85_base$RCP85_Upper_95_CI,
                             rcp45_per1$RCP45_Upper_95_CI,
                             rcp85_per1$RCP45_Upper_95_CI)
                        )
        )

legend("topleft",
       fill   = c("blue", "red"),
       legend = c("RCP 4.5", "RCP 8.5"))
```



Rapid City LOCA Return Intervals for 2020−2049
RCP 4.5
RCP 8.5
Maximum Total Daily Precip (mm)
40
60
80
100
120
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Return Period (Years)